Compressing Forwarding Tables for Datacenter Scalability

Ori Rottenstreich, Marat Radan, Yuval Cassuto, Isaac Keslassy,
Carmi Arad, Tal Mizrahi, Yoram Revah and Avinatan Hassidim

***Abstract*—With the rise of datacenter virtualization, the number of entries in the forwarding tables of datacenter switches is expected to scale from several thousands to several millions. Unfortunately, such forwarding table sizes would not fit on-chip memory using current implementations.**

In this paper, we investigate the compressibility of forwarding tables. We first introduce a novel forwarding table architecture with separate encoding in each column. It is designed to keep supporting fast random accesses and fixed-width memory words. Then, we show that although finding the optimal encoding is NP-hard, we can suggest an encoding whose memory requirement per row entry is guaranteed to be within a small additive constant of the optimum. Next, we analyze the common case of two-column forwarding tables, and show that such tables can be presented as bipartite graphs. We deduce graph-theoretical bounds on the encoding size. We also introduce an algorithm for optimal conditional encoding of the second column given an encoding of the first one. In addition, we explain how our architecture can handle table updates. Last, we evaluate our suggested encoding techniques on synthetic forwarding tables as well as on real-life tables.

***Index Terms*—Datacenter Virtualization, Layer-2 Datacenter, Forwarding Information Base, Compression.**

I. Introduction

A. Background

The rapid growth of forwarding tables in network switches raises serious concerns in the networking industry. Layer 2 (L2) networks are no longer constrained to small local area networks. On the contrary, they are now used in datacenter networks, which will soon have a need for millions of hosts in a single L2 domain, while current L2 domains are only restricted to several thousands [2]. As a result, current forwarding tables cannot handle such datacenter networks.

The main driver of this expected dramatic growth rate is the deployment of host *virtualization* in datacenter networks. Whereas traditional servers would only use one or two globally-unique MAC addresses per server, contemporary servers use tens of MAC addresses, corresponding to their tens of virtual machines (these generated MAC addresses are not necessarily globally unique anymore). Assuming one MAC address per thread, and considering the joint increase of the numbers of (a) threads per core, (b) cores per processor (socket), (c) sockets per server, and (d) servers per datacenter, it is clear that the product of all these numbers is expected to dramatically increase in the coming years [2]–[4]. In addition to this expected strong growth in the number of entries per forwarding table, we can also expect a moderate growth in the number of attributes per entry, such as the quality-of-service and security attributes, as well as in the size of these attributes. All in all, we need to be able to handle dramatically larger forwarding tables.

Unfortunately, forwarding tables must be accessed on a per-packet basis. Therefore, they are performance sensitive. Because of the high datacenter rates, they are often implemented in hardware using *on-chip memory*. This makes scaling forwarding tables challenging, because on-chip memory is too small to hold millions of forwarding-table entries in raw form [5], [6].

The goal of this paper is to study the compressibility of forwarding tables, providing both an analytical framework and experimental insight toward the effective compression of such tables.

B. Compressing Forwarding Tables

Forwarding tables, also referred to as Forwarding Information Bases (FIBs), consist of several entries, which associate a *lookup key* to its corresponding *forwarding information*. For instance, the lookup key can be a (*destination MAC address*, *VLAN*) pair, and the forwarding information can include the switch target port, the quality-of-service attributes, the security attributes, and additional information used for various modifications that need to be applied to the packet.

Forwarding tables are typically arranged as *two-dimensional tables*. Each row entry in a forwarding table corresponds to a different lookup key, and contains several columns, with the lookup key and forwarding information. Forwarding tables can then be accessed using a hashing mechanism. Upon each incoming packet, the packet lookup key is hashed into one (or several) of the hash table entries. If this entry indeed contains the lookup key, then the forwarding table returns the associated forwarding information.

Forwarding tables are typically redundant, because their fields (columns) can have a highly non-uniform value distribution, and because the values in different fields can be correlated. Therefore, we are interested in studying whether

Manuscript received January 15, 2013; revised May 28, 2013; accepted July 1, 2013.

This manuscript is an extended version of "Compressing forwarding tables", which was presented in IEEE Infocom '13, Turin, Italy, April 2013 [1].

O. Rottenstreich, M. Radan, Y. Cassuto and I. Keslassy are with the Technion, Israel (e-mails: {or@tx, radan@tx, ycassuto@ee, isaac@ee}.technion.ac.il).

C. Arad, T. Mizrahi and Y. Revah are with Marvell Israel (e-mails: {carmi, talmi, yoramr}@marvell.com). T. Mizrahi is also with the Technion.

A. Hassidim is with the department of computer science, Bar-Ilan University, Israel (e-mail: avinatan@cs.biu.ac.il).

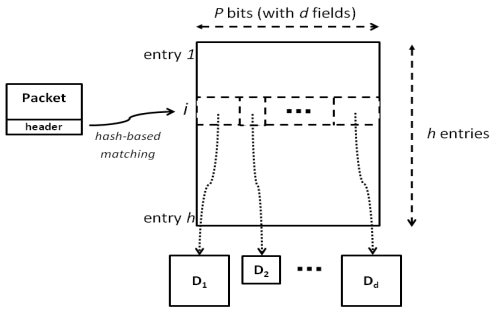


Fig. 1. An illustration of an access to the encoded forwarding table. First, the packet lookup key is hashed to obtain an entry index $i \in [1, h]$. Then, a single encoded entry of (at most) P bits is retrieved. Finally, d pipelined accesses to d dictionaries are performed based on the d binary codewords (of different lengths) composing this encoded entry.

these properties can be used to compress them. However, we would like to keep two key properties that are specific to forwarding tables:

(i) The ability to access each row entry directly regardless of its row index, and

(ii) The ability to store each row entry in a fixed-width memory word.

Such criteria prevent or make less attractive traditional encoding techniques. For instance, if the entries are compressed using a Lempel-ZivWelch (LZW) technique and sequentially stored in the memory, it is then difficult to calculate the memory address of a specific entry, and to access it directly based on its entry index.

C. Overview of the Suggested Solution

As illustrated in Figure 1, we suggest a *novel compression technique for forwarding tables* that supports random accesses.

We propose to encode each of the forwarding-table columns separately, using a dedicated variable-length *binary prefix encoding*, i.e. an encoding in which any codeword is not a prefix of any other codeword. We remind that a prefix of a string $b_1 b_2 \dots b_n$, is a string of the form $b_1 b_2 \dots b_m$ where $m \leq n$. Therefore, each row entry in a d-column table is represented by the concatenation of d codewords representing the content of each of its d fields. Accordingly, its width equals the sum of the d corresponding codeword lengths. For each row entry we allocate a fixed memory size P. In addition, each column j is associated to a dictionary D_j, which is stored separately in the memory. This dictionary D_j maps each binary codeword into its represented field value in column j.

Figure 1 illustrates an access to the encoded forwarding table. At each packet arrival, we first hash the packet lookup key to obtain an entry index $i \in [1, h]$. Then, we access the encoded table to retrieve the P bits representing the encoding of its i^{th} entry. Last, we look up each of the d dictionaries in a pipelined manner. This way, we sequentially obtain the corresponding values of the d fields in the i^{th} entry after d pipelined lookups.

Note that the combination of the bounded-width entry property and the prefix property of individual fields yields our two desired properties specified above.

(A) The original forwarding table

MAC Address	VLAN	Target Port
00:1b:2b:c3:4d:90	Vlan10	Te12/1
00:00:aa:6c:b1:10	Vlan10	Gi11/8
00:00:aa:65:ce:e4	Vlan10	Te12/1
00:00:aa:65:ce:e4	Vlan200	Gi11/24
00:13:72:a2:a2:0e	Vlan200	Gi11/24
00:21:9b:37:7e:14	Vlan10	Te12/1
00:13:72:a2:a2:0e	Vlan200	Gi11/8

(B) Encoding

110	0	0
00	0	10
01	0	0
01	1	11
10	1	11
111	0	0
10	1	10

(C) The encoding dictionaries

D_1 (MAC Address)	D_2 (VLAN)	D_3 (Target Port)
00 - 00:00:aa:6c:b1:10	0 - Vlan10	0 - Te12/1
01 - 00:00:aa:65:ce:e4	1 - Vlan200	10 - Gi11/8
10 - 00:13:72:a2:a2:0e		11 - Gi11/24
110 - 00:1b:2b:c3:4d:90		
111 - 00:21:9b:37:7e:14		

TABLE I

AN EXAMPLE OF A FORWARDING TABLE WITH $d = 3$ COLUMNS AND ITS ENCODED REPRESENTATION. (A) SHOWS THE ORIGINAL TABLE WITHOUT ENCODING. THE ENTRY SELECTION MECHANISM IS NOT PRESENTED. (B) ILLUSTRATES THE CORRESPONDING ENCODED FORWARDING TABLE WITH A MAXIMAL ENCODED ENTRY WIDTH OF $P = 5$ BITS. (SPACES ARE PRESENTED JUST FOR SIMPLICITY AND DO NOT EXIST IN PRACTICE.) (C) DESCRIBES THE $d = 3$ DICTIONARIES REPRESENTING THE PREFIX ENCODINGS OF THE d COLUMNS.

Example 1. *Table I presents an example of a forwarding table and a possible encoding of that table. There are $h = 7$ entries in the table, each with $d = 3$ fields: MAC Address, VLAN and Target Port.*

First, Table (A) presents the original table without encoding.

Next, Table (B) illustrates a possible encoding of the forwarding table (the presented spaces in the encoded table are just shown to distinguish between the different fields, and do not exist in practice). For instance, the two possible values in the VLAN field are encoded by a simple fixed-length single-bit encoding. In addition, variable-length encodings are used to encode the three distinct values in the Target Port field, as well as the five distinct values of the MAC Address field. With these $d = 3$ encodings, the obtained maximal width of an encoded entry is $P = 5$ bits. Thus, in order to support the random-access requirement, we keep $P = 5$ bits in the memory for each encoded entry, even if it may actually be shorter. For instance, the third entry only needs 4 bits, and therefore wastes 1 bit.

Finally, Table (C) illustrates the encodings used by the $d = 3$ dictionaries representing the $d = 3$ columns. The number of entries in each dictionary equals the number of distinct values in the corresponding column in (A).

This example illustrates how prefix encoding enables us to simply *concatenate the codewords* without a need for any additional separating flags.

In addition, it also shows how the memory size is directly determined by the *worst-case entry width*, i.e. the maximum number of concatenated bits per entry. This is one reason for example why encoding each column using *Huffman coding* [7] would not necessarily address this problem: while it is optimal for the *average* entry width, it may yield a worst case that ends up wasting a significant number of bits. Therefore, for a given

forwarding table with d columns, *our goal is to suggest a set of d prefix encodings that minimizes the maximal encoded entry width.* This makes it particularly hard, because each entry width involves *all columns*, and the maximal one also involves *all row entries*. Therefore, any strong solution may need to *jointly* look at all the elements.

D. Related Work

The forwarding table scaling problem has been thoroughly analyzed and discussed in networking industry forums and standards organizations, both in the L2 and the L3 layers (e.g. [2], [4]–[6], [8], [9]). These analyses raise the concern that the steep growth rate of forwarding tables has outrun the steady technology growth rate of memory density.

Various creative methods have been proposed to mitigate this scaling problem. First, *FIB aggregation* [10] is an approach that reduces the size of a forwarding table by aggregating table entries with the same values of the next hop field. For instance, two prefix rules that differ only on their last bit can be combined to a single prefix with a shorter prefix. Likewise, a prefix may be eliminated due to redundancy achieved by a shorter prefix with the same next hop. A similar approach is presented in [11]–[14]. For instance, in [12] an optimal representation is found based on dynamic programming where in each rule the value of the next hop can be selected among a short list of possible values. However, this approach assumes that the routing table has a tree structure, i.e. the rules have the form of prefixes in an address field (usually IP address) and cannot be applied in the case of more general fields, such as an exact MAC address with a corresponding attribute.

In addition, there have been studies in other related fields. Clearly, in *information theory*, there are many results on compression and on lower bounds based on entropy [7]. Likewise, *database implementations* also attempt to scale systems by introducing compression techniques [15]–[17]. In particular, in [15] a dictionary-based solution that supports delete and update operations is suggested. A field value is inserted into the dictionary and encoded based on its length and its number of occurrences. Likewise, a block of data is compressed only if its compression ratio is beyond a threshold. However, these studies do not consider the forwarding table restrictions that we mention above, and in particular the bounded entry length. Recently, [18] studied how to maximize the probability of encoding random entries within a fixed-width memory.

Our solution can be considered as a type of topological transformation. Most of the related approaches suggest TCAM-based solutions and mainly discuss range encoding [19]–[21]. Our solution can be applied to a more general class of tables and is not necessarily based on TCAM.

Finally, as mentioned, an orthogonal and complementary consideration is the implementation of the forwarding table using hashing, in order to easily find the row that corresponds to a given key [22].

E. Contributions

This paper investigates a novel technique for the efficient representation of forwarding tables in datacenter switches, with a support for fast random access and fixed-width memory words.

We start by studying an offline encoding scheme for a given static FIB table. We prove that it is possible to formulate the encoding problem as an optimization problem, and that the relaxed optimization problem is convex. As a result, we propose a novel algorithm that achieves, for any given forwarding table with d columns, the optimal worst-case entry width *within a guaranteed constant additive factor of* $d-1$, independently of their values and of the number of rows.

Next, we also prove that the general optimization problem is *NP-hard*, as long as there are at least $d = 7$ columns. We do so by presenting a reduction from the $3-SAT$ decision problem. Therefore, there is little hope for a practical optimal algorithm.

In addition, we also consider the special case of forwarding tables with $d = 2$ columns, which may only consist of the key and the action. We first suggest tight upper and lower bounds on the optimal encoding width of a two-column table in the general case. We also show a sufficient condition for a fixed-length encoding to be optimal. Moreover, we present an *optimal conditional encoding* of a second column when the encoding of one column is given. Later, we also introduce a bipartite graph that can model the forwarding table, and present additional graph-theoretic bounds on its optimal encoding width. These improved bounds rely on the combinatorial properties of the graph, such as the size of its edge cover.

Finally, we also discuss the limits of our offline model, by analyzing how our compressed forwarding tables may be able to handle *updates*. We show that an additional bit per column and an additional small memory may be sufficient to cope with temporary loads of entry additions, modifications or deletions.

We conclude by evaluating the suggested encoding techniques on real-life forwarding tables as well as on synthetic tables. We demonstrate how our introduced techniques fare against alternative algorithms such as fixed-length and Huffman coding. In particular, we get closer to the compression lower bound, albeit at the cost of an increased complexity.

II. Model and Problem Formulation

A. Terminology

We start with the formal definition of terminology used in this paper. For a binary codeword (string) x, let $\ell(x)$ denote the length in bits of x.

Definition 1 (FIB Table)**.** *A* FIB table $A = ((a_1^1,\ldots,a_d^1),\ldots,(a_1^h,\ldots,a_d^h))$ *is a two-dimensional matrix of h rows (entries) and d columns (fields).*

Given a FIB table A, we denote by S_j (for $j \in [1,d]$) the set of element values in the j^{th} column, i.e. $S_j = \bigcup_{i=1}^{h}\{a_j^i\}$. We also denote by n_j the number of distinct elements in the j^{th} column, i.e. $n_j = |S_j|$. Let $s_{j,1},\ldots,s_{j,n_j}$ denote the elements of S_j.

For example, in Table I, there are $n_3 = 3$ values in the third column, and $S_3 = \{$Te12/1,Gi11/8, Gi11/24$\}$.

Definition 2 (Prefix Encoding)**.** *For a set of elements S, an encoding σ is an injective mapping $\sigma : S \to B$, where B is a set of binary codewords of size $|B| = |S|$. An encoding is called a* prefix encoding *if no binary codeword in B is a prefix (start) of any other binary codeword in B.*

Definition 3 (FIB Encoding Scheme)**.** *An* encoding scheme *C_A of a FIB Table A is an ordered set of d* prefix *encodings $C_A = (\sigma_1, \ldots, \sigma_d)$. That is, each σ_j encodes column j, and is constrained to be a prefix encoding.*

Definition 4 (FIB Encoding Width)**.** *Let $C_A = (\sigma_1, \ldots, \sigma_d)$ be a FIB encoding scheme of the FIB table A. The encoding width $\ell(C_A)$ of C_A is defined as the maximal sum of the lengths of the d codewords representing the d elements in a row of A, i.e.*

$$\ell(C_A) = \max_{i \in [1,h]} \left(\sum_{j=1}^{d} \ell(\sigma_j(a_j^i)) \right). \tag{1}$$

Example 2. *Let A be the FIB table from Example 1 with $h = 7$, $d = 3$, also described in Table I.(A). Likewise, let $C_A = (\sigma_1, \sigma_2, \sigma_3)$ be the FIB encoding scheme of A. Then, as shown in Table I.(C), σ_3(Te12/1) = 0, σ_3(Gi11/8) = 10, σ_3(Gi11/24) = 11. As described in Table I.(B), $\ell(C_A) = \max_{i \in [1,h]} \left(\sum_{j=1}^{d} \ell(\sigma_j(a_j^i)) \right) = \max\left(5, 5, 4, 5, 5, 5, 5\right) = 5$.*

Let's compare this scheme with a fixed-length encoding of each column. Since the number of distinct elements in the $d = 3$ columns of A are $n_1 = 5, n_2 = 2$ and $n_3 = 3$, if I_A a fixed-length encoding of A then the elements in the j^{th} column are encoded in $\lceil \log_2 n_j \rceil$ bits, and $\ell(I_A) = \sum_{j=1}^{d} \lceil \log_2 n_j \rceil = 3 + 1 + 2 = 6$.

B. Optimal FIB Table Encoding Scheme

For each FIB table A, we denote by $OPT(A)$ the optimal encoding width of A, i.e. the smallest possible encoding width of any encoding scheme of A such that

$$OPT(A) = \min_{C_A = (\sigma_1, \ldots, \sigma_d)} \ell(C_A). \tag{2}$$

Our goal is to find an encoding scheme C_A that minimizes the encoding width $\ell(C_A)$, and therefore reaches this minimum.

III. General FIB Tables

A. Optimization Problem

We first want to rewrite the problem of finding the optimal encoding scheme for any FIB table A as an optimization problem.

First, it is well known that the set of codeword lengths in a prefix encoding exists iff it satisfies *Kraft's inequality* [7].

Property 1 (Kraft's inequality)**.** *There exists a prefix encoding σ of the elements in a set S with codeword lengths $\{\ell(\sigma(a)) | a \in S\}$ iff*

$$\sum_{a \in S} 2^{-\ell(\sigma(a))} \leq 1. \tag{3}$$

Therefore, we can now formally present the problem of finding an optimal FIB table encoding scheme for table $A = ((a_1^1, \ldots, a_d^1), \ldots, (a_1^h, \ldots, a_d^h))$ as the following optimization problem.

$$\begin{aligned}
\min \quad & P \\
\text{s.t.} \quad & \sum_{j=1}^{d} \ell(\sigma_j(a_j^i)) \leq P && \forall i \in [1,h] && \text{(4a)} \\
& \sum_{a \in S_j} 2^{-\ell(\sigma_j(a))} \leq 1 && \forall j \in [1,d] && \text{(4b)} \\
& \ell(\sigma_j(a)) \geq 0 \ \forall j \in [1,d], && \forall a \in S_j && \text{(4c)} \\
& \ell(\sigma_j(a)) \in \mathbb{Z} \ \forall j \in [1,d], && \forall a \in S_j && \text{(4d)}
\end{aligned}$$

In this optimization problem, we try to minimize the maximal encoding width of a row (denoted here by P) while having four sets of constraints. The first set (4a) represents the limitation on the total width of each row. The second set of constraints (4b) requires that each of the encodings $\sigma_1, \ldots, \sigma_d$ should satisfy Kraft's inequality. The last two sets (4c, 4d) illustrate the fact that the codeword length of any element should be a non-negative integer (the constraints (4b) and (4c) together guarantee that if the number of distinct elements in a column is at least two, the codeword lengths are positive). For an optimal solution to this optimization problem, the equality $P = OPT(A)$ is satisfied.

The optimization problem can be also described as being dependent only on the codeword lengths. To do so, we replace (for $j \in [1,d], a \in S_j$) the value of $\ell(\sigma_j(a))$ by a corresponding variable. In addition, it is easy to derive the sets of codewords given their lengths. To do so, we can find for an encoding σ_i (for $i \in [1,d]$), the codewords of the elements of S_i in an increasing order of their required lengths. For each element $x \in S_i$, we allocate an arbitrary codeword $\sigma_i(x)$ in the required length such that none of the previously allocated codewords (for this encoding) is a prefix of the current codeword. It is easy to verify that Kraft's inequality guarantees that such allocation is always possible.

We also denote by the *relaxed FIB encoding problem* the problem achieved by omitting the fourth set of constraints (4d). In this relaxed problem, the codeword lengths are not necessarily integers.

B. Approximation of the Optimal Encoding

We now describe how to obtain a concrete efficient encoding scheme for any given forwarding table with an arbitrary number of columns. We find for each table A of d columns, a solution to the FIB encoding problem that is guaranteed to be within a *fixed additive approximation* of the optimal encoding width $OPT(A)$. More specifically, we find an encoding scheme C_A with encoding width $\ell(C_A)$ that satisfies $\ell(C_A) \leq OPT(A) + (d-1)$, i.e. its encoding width is larger than the optimal encoding width (in bits) by at most the number of columns in A minus one. We emphasize that this bound on the additive error of $(d-1)$ depends neither on the number of distinct elements $n_j \leq 2^{W_j}$ (for $j \in [1,d]$) in each of the columns, nor on the number of rows h in A.

To obtain such an encoding, we consider the *relaxed FIB encoding problem* defined above, in which the codeword lengths are not necessarily integers. We show that this optimization problem is *convex*, and thus its solution can be found by one of several known algorithms for such problems. We then build as a solution to the original optimization problem, an encoding with codeword lengths achieved by taking the ceiling of each of the codeword lengths in the solution of the relaxed problem. We show that these new codeword lengths lead to an encoding that satisfies the additive approximation from above.

Theorem 1. *Let* $\big(\ell(\bar{\sigma}_j(s_{j,1})),\ldots,\ell(\bar{\sigma}_j(s_{j,n_j}))\big)$ *(for* $j \in [1,d]$*) be the codeword lengths of an optimal solution to the relaxed FIB encoding problem. Further, let* $C_A=(\sigma_1,\ldots,\sigma_d)$ *be an encoding scheme satisfying* $\ell(\sigma_j(s_{j,i}))=\lceil \ell(\bar{\sigma}_j(s_{j,i}))\rceil$ *for all* $j\in[1,d], i\in[1,n_j]$*. Then we have:*
(i) The relaxed FIB encoding problem is convex.
(ii) The encoding width $\ell(C_A)$ *of the encoding scheme* C_A *satisfies*

$$\ell(C_A) \leq OPT(A) + (d-1). \tag{5}$$

Proof: We first examine the convexity of the relaxed problem. Clearly, its simple objective function is convex. We would like to show that each of the inequality constraint functions (as a function of the codeword lengths) is convex as well. Simply, constraints in the first and the third sets of inequality constraints (4a, 4c) are convex due to their linearity. In addition, we would now like to examine the convexity of the second set of constraints (4b) representing Kraft's inequality. To do so, we define d functions $f_1,\ldots,f_d$ for the constraints on each of the d prefix encodings. The function f_j (for $j\in[1,d]$) receives as a parameter the set of codeword lengths of the elements in S_j. The function $f_j\big(\ell(\sigma_j(s_{j,1})),\ldots,\ell(\sigma_j(s_{j,n_j}))\big)$ is defined as $\sum_{a\in S_j} 2^{-\ell(\sigma_j(a))} = \sum_{i=1}^{n_j} 2^{-\ell(\sigma_j(s_{j,i}))}$.

We consider two arbitrary encoding schemes $(\bar{\sigma}_1,\ldots,\bar{\sigma}_d),(\hat{\sigma}_1,\ldots,\hat{\sigma}_d)$ with codeword lengths $\big(\ell(\bar{\sigma}_j(s_{j,1})),\ldots,\ell(\bar{\sigma}_j(s_{j,n_j}))\big)$, $\big(\ell(\hat{\sigma}_j(s_{j,1})),\ldots,\ell(\hat{\sigma}_j(s_{j,n_j}))\big)$ for the elements in S_j for $j\in[1,d]$, respectively. We would like to show that $f_j(\alpha\cdot\bar{\sigma}+\beta\cdot\hat{\sigma}) \leq \alpha\cdot f_j(\bar{\sigma})+\beta\cdot f_j(\hat{\sigma})$ for all $j\in[1,d]$ and $\alpha,\beta\in[0,1]$ with $\alpha+\beta=1$. Here, when performing linear operations on the set of codeword lengths, we refer to a new set of codeword lengths, so that each of its codeword lengths is obtained by performing the linear operations on the corresponding length in the original set. Based on the convexity of the function $g(x)=2^{-x}$, we have

$$\begin{aligned}
f_j(\alpha\cdot\bar{\sigma}+\beta\cdot\hat{\sigma}) &= \sum_{i=1}^{n_j} 2^{-\left(\alpha\cdot\ell(\bar{\sigma}_j(s_{j,i}))+\beta\cdot\ell(\hat{\sigma}_j(s_{j,i}))\right)} \\
&\leq \sum_{i=1}^{n_j}\left(\alpha\cdot 2^{-\ell(\bar{\sigma}_j(s_{j,i}))}+\beta\cdot 2^{-\ell(\hat{\sigma}_j(s_{j,i}))}\right) \\
&= \alpha\cdot\sum_{i=1}^{n_j} 2^{-\ell(\bar{\sigma}_j(s_{j,i}))}+\beta\cdot\sum_{i=1}^{n_j} 2^{-\ell(\hat{\sigma}_j(s_{j,i}))} \\
&= \alpha\cdot f_j(\bar{\sigma})+\beta\cdot f_j(\hat{\sigma}).
\end{aligned} \tag{6}$$

Based on these properties, we can finally deduce that the relaxed FIB encoding problem is convex. With this observation, an optimal solution, i.e. a generalized encoding (with non-necessarily integer codeword lengths) can be found by using one of the several known (polynomial time) algorithms for such convex problems (e.g. the ellipsoid algorithm [23]). Let us denote by

$$\big(\ell(\bar{\sigma}_j(s_{j,1})),\ldots,\ell(\bar{\sigma}_j(s_{j,n_j}))\big)$$

the codeword lengths of an optimal solution to the relaxed problem. Let $OPT_r(A)=\max_{i\in[1,h]}\left(\sum_{j=1}^{d}\ell(\bar{\sigma}_j(a_j^i))\right)$ be the maximal encoding width of a row in this optimal solution of the relaxed problem. Again, $OPT_r(A)$ is not necessarily integer. Clearly, since the set of constraints in the relaxed problem is a subset of the constraints in the FIB encoding problem, the inequality $OPT_r(A)\leq OPT(A)$ holds. Further, since $OPT(A)$ is an integer, we have that $\lceil OPT_r(A)\rceil \leq OPT(A)$, i.e. $\lceil OPT_r(A)\rceil$ *is a lower bound for* $OPT(A)$. Clearly, since $\ell(\sigma_j(s_{j,i}))=\lceil\ell(\bar{\sigma}_j(s_{j,i}))\rceil\geq\ell(\bar{\sigma}_j(s_{j,i}))$, all the d constraints representing Kraft's inequality, satisfied by the solution of the relaxed problem, are also satisfied by the set of codeword lengths in $C_A=(\sigma_1,\ldots,\sigma_d)$.

We now show that the encoding width of the encoding scheme C_A is within a guaranteed additive approximation of the optimal encoding width of A, $OPT(A)$. Directly from the definition of C_A, we can have that

$$\begin{aligned}
\ell(C_A) &= \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\ell(\sigma_j(a_j^i))\right) = \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\lceil\ell(\bar{\sigma}_j(a_j^i))\rceil\right) \\
&< \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\left(\ell(\bar{\sigma}_j(a_j^i))+1\right)\right) \\
&= \max_{i\in[1,h]}\left(\sum_{j=1}^{d}\ell(\bar{\sigma}_j(a_j^i))\right)+d = OPT_r(A)+d \\
&\leq OPT(A)+d.
\end{aligned} \tag{7}$$

Finally, since $\ell(C_A)$ as well as $OPT(A)$ are both integers, we can deduce from the (strong) inequality $\ell(C_A)<OPT(A)+d$ that $\ell(C_A)\leq OPT(A)+(d-1)$. ∎

C. Upper Bound on the Optimal Encoding

We now present a simple upper bound on the optimal encoding of any FIB table with an arbitrary number d of columns.

Consider a fixed-length encoding scheme I_A that encodes each column j using a fixed-length encoding (as seen in Example 2). Then in each column j, it uses codewords of size $W_j=\lceil\log_2 n_j\rceil$, since it needs at least $n_j>2^{W_j-1}$ codewords to represent the n_j different values. As a consequence, the total width of each row in also fixed, and we have

$$\ell(I_A)=\sum_{j=1}^{d}W_j=\sum_{j=1}^{d}\lceil\log_2 n_j\rceil. \tag{8}$$

The fixed-length encoding yields an upper bound on the optimal encoding width of the FIB table.

Property 2. *Let A be a FIB table of d columns such that the set of distinct elements in the j^{th} column is S_j with $|S_j| = n_j \leq 2^{W_j}$. Then,*

$$OPT(A) \leq \sum_{j=1}^{d} W_j. \tag{9}$$

D. NP-HARDNESS

We conclude the section by considering the complexity of finding an optimal encoding scheme of a given FIB table. First, for a table with a single column ($d = 1$) an optimal solution is given by the fixed-length encoding. If there are $n_1 = |S_1|$ distinct elements, in the fixed-length encoding all codewords have length of $\lceil \log_2 n_1 \rceil$ bits. Clearly, this encoding is optimal since Kraft's inequality cannot be satisfied if all elements are encoded in at most ($\lceil \log_2 n_1 \rceil$-1) bits.

We now show that the above encoding problem is NP-hard for tables with at least seven columns.

Theorem 2. *Given a FIB table A with $d \geq 7$ columns, finding an optimal encoding scheme of A is* NP-hard.

Proof Outline: For space reasons, we provide a proof outline, and present the full proof in [24]. We show a reduction from the $3 - SAT$ decision problem to finding an optimal encoding of a table with 7 columns.

Let $Y = Y_1 \wedge Y_2 \wedge \ldots \wedge Y_m$ be a CNF formula. Let n be the number of variables in Y. For simplicity, we assume that n is of the form of $n = \frac{2^k-1}{3}$ for a positive integer k.

We show that the formula Y is satisfiable iff a carefully constructed table $T(Y)$ with seven columns has encoding width of at most $3k + 6$ bits. $T(Y)$ has two kinds of entries. Entries that are not influenced by Y, and m additional entries that represent the m clauses of Y. A clause is satisfied if the width of the corresponding encoded entry is at most $3k+6$ bits. Using this table, we present the reduction from any instance of $3 - SAT$, before concluding. ∎

Determining the hardness of the problem in the case of $d \in [2, 6]$ is left as an open question. We conjecture that the problem is NP-hard for $d \geq 2$.

Conjecture 1. *For all $d \geq 2$, finding an optimal encoding scheme of tables with d columns is* NP-hard.

E. Retrieving the Forwarding Information

As mentioned in Section I, in the proposed solution the representation of any encoded entry of d columns, is simply the concatenation of the d corresponding codewords without any additional separating flags. We would like to show first that the representations of two distinct original entries yield two different concatenations. We would also like to suggest an efficient way to detect the boundaries of the different codewords composing such a concatenation.

Property 3. *Let $\sigma_1, \ldots, \sigma_d$ be d prefix encodings of d sets $S_1, \ldots, S_d$. Let $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$ be two entries satisfying $(\forall j \in [1, d])x_j, y_j \in S_j$. Let $\sigma(x) = \sigma_1(x_1) \cdot \sigma_2(x_2) \ldots \sigma_d(x_d)$ and $\sigma(y) = \sigma_1(y_1) \cdot \sigma_2(y_2) \ldots \sigma_d(y_d)$ be the two string obtained by concatenating the d codewords in the encodings of x, y. If $\sigma(x) = \sigma(y)$ then necessarily $x = y$, i.e. $(\forall j \in [1, d])x_i = y_i$. In particular there is a single element $z_1 \in S_1$ whose encoding $\sigma_1(z_1)$ is a prefix of $\sigma(x) = \sigma(y)$.*

Proof: The proof is by induction on d. First, for $d = 1$ if $\sigma(x) = \sigma(y)$ we have $\sigma_1(x_1) = \sigma(x) = \sigma(y) = \sigma_1(y_1)$. Since σ_1 is an encoding, we have that necessarily $x_1 = y_1$. For $d \geq 2$, assume that $\sigma(x) = \sigma(y)$, i.e. $\sigma_1(x_1) \cdot \sigma_2(x_2) \ldots \sigma_d(x_d) = \sigma_1(y_1) \cdot \sigma_2(y_2) \ldots \sigma_d(y_d)$ and distinguish between the two following cases.
(i) If $x_1 = y_1$ then necessarily $\sigma_1(x_1) = \sigma_1(y_1)$. We can then deduce that $\sigma_2(x_2) \ldots \sigma_d(x_d) = \sigma_2(y_2) \ldots \sigma_d(y_d)$. By the induction hypothesis we have in this case also that $(x_2, \ldots, x_d) = (y_2, \ldots, y_d)$ and since $x_1 = y_1$ we can conclude that $(x_1, \ldots, x_d) = (y_1, \ldots, y_d)$.
(ii) If $x_1 \neq y_1$ then necessarily $\sigma_1(x_1) \neq \sigma_1(y_1)$. From the equality $\sigma_1(x_1) \cdot \sigma_2(x_2) \ldots \sigma_d(x_d) = \sigma_1(y_1) \cdot \sigma_2(y_2) \ldots \sigma_d(y_d)$ we must have that $\ell(\sigma_1(x_1)) \neq \ell(\sigma_1(y_1))$ and that either $\sigma_1(x_1)$ is a prefix of $\sigma_1(y_1)$ (if $\ell(\sigma_1(x_1)) < \ell(\sigma_1(y_1))$) or $\sigma_1(y_1)$ is a prefix of $\sigma_1(x_1)$ (if $\ell(\sigma_1(x_1)) > \ell(\sigma_1(y_1))$). Both cases lead to a contradiction since σ_1 is a prefix encoding.

Finally, since case (ii) leads to a contradiction, the only element $z_1 \in S_1$ whose encoding $\sigma_1(z_1)$ is a prefix of $\sigma(x) = \sigma(y)$ must be $z_1 = x_1 = y_1$. ∎

Following the last part of the property, we can now suggest how to detect the boundaries of the different codewords composing an encoded entry. To do so, we keep for each of the d encodings, a dictionary with a sorted list of the possible codewords (and their matching elements). Then, we perform in a pipelined manner d accesses to the dictionaries. In each access, we perform a binary search to find the single codeword in the current encoding which is a prefix of the concatenated string representing the encoded entry. By the last property, there is exactly one such codeword. Next, we continue to the next dictionary after omitting the prefix of the concatenated string. Any additional memory besides the basic representation of the dictionaries is not required.

IV. TWO-COLUMN FIB TABLES

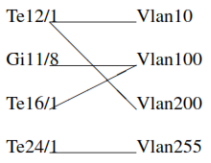
In datacenters, a popular class of FIB tables are simple L2 MAC tables that contain two columns. The first describes the Target Port, while the second can be viewed as an aggregation of columns representing a collection of other attributes. Indeed, we see in real-life table traces that these additional attributes are hardly ever used. Thus their aggregation has a relatively small set of possible values.

Therefore, from now on, we consider the special case of two-column FIB tables, i.e. FIB tables that satisfy $d = 2$. We would like to fundamentally study this case to obtain some intuition on the problem. We show that in this case, in order to find its optimal encoding scheme, a FIB table A can be simply represented as a bipartite graph. We also suggest an analysis of the optimal encoding width of such two-column FIB tables.

For the sake of simplicity, we assume, unless mentioned otherwise, that the number of distinct elements in each of the $d = 2$ columns of A is the same, and that it is a power of two, i.e. $|S_1| = |S_2| = n = 2^W$.

Target Port	Interface
Te12/1	Vlan10
Gi11/8	Vlan100
Te12/1	Vlan200
Te16/1	Vlan100
Te24/1	Vlan255
Te16/1	Vlan100

(a)



(b)

Fig. 2. A two-column forwarding table with $h = 6$ entries and $n = 2^W = 4$ (for $W = 2$) distinct elements in each column with (a) the table, and (b) its corresponding bipartite graph. The number of edges in the graph equals the number of distinct entries in the forwarding table.

A. The Optimal Encoding Width of a Two-Column FIB Table

The next theorem suggests a lower bound and an upper bound on the optimal encoding width of $A = ((a_1^1, a_2^1), \ldots, (a_1^h, a_2^h))$ for $W \geq 2$.

Theorem 3. *Let A be a two-column FIB table with $|S_1| = |S_2| = n = 2^W$ for $W \geq 2$. Then, the optimal encoding width of A satisfies*

$$W + 2 \leq OPT(A) \leq 2W. \tag{10}$$

Proof Outline: In [24], we present the full proof and also prove the tightness of these bounds by exposing, for each of them, a FIB table that achieves the bound.

The upper bound of $W + W = 2W$ can be simply achieved using Property 2, based on a fixed-length encoding. To show the lower bound, we consider two options for the codeword lengths of the elements in S_1. If all of them equal W bits, at least one of them shares an entry with an element in S_2 encoded with two or more bits. Likewise, if the codeword lengths are not fixed, we have an element in S_1 with a codeword length of at least $W + 1$ that shares an entry with another arbitrary element in S_2 encoded in at least a single bit. Both cases then suffice to conclude. ∎

Any two-column FIB table $A = ((a_1^1, a_2^1), \ldots, (a_1^h, a_2^h))$ can be represented by a corresponding bipartite graph G_A as follows. The two disjoint sets are the sets of distinct elements in each of the two columns. Thus, if an element appears in both of the two columns, it is represented by two different vertices in each of the two disjoint sets. Edges connect elements in the two sets if they appear at least once on the same row of the FIB table. Formally, we define the graph $G_A =< L + R, E >$ such that $L = S_1$, $R = S_2$ and $E = \{(x, y) | (\exists i \in [1, h]), (a_1^i, a_2^i) = (x, y)\}$. Therefore, duplicated rows have no influence on the construction of the bipartite graph, and the graph does not contain parallel edges. It is also easy to see the independence in the order of the rows of the FIB table.

Example 3. *Figure 2(a) presents an example of a two-column forwarding table with $h = 6$ entries and $n = 2^W = 4$ (for $W = 2$) distinct elements in both columns. The corresponding bipartite graph appears in Figure 2(b). The vertices on the left side of the graph represent the $n = 4$ distinct elements in the first column, while the vertices on the right side represent the n distinct elements in the second column. The number of edges in the graph equals the number of distinct entries in the forwarding table.*

Given a FIB encoding scheme $C_A = (\sigma_1, \ldots, \sigma_d)$ of $A = ((a_1^1, a_2^1), \ldots, (a_1^h, a_2^h))$, we can present its FIB encoding width based on G_A as $\ell(C_A) = \max_{(x,y) \in E} \ell(\sigma_1(x)) + \ell(\sigma_2(y))$. From the construction of G_A, we can clearly see that the last equation is compatible with Definition 4.

The representation of a FIB table as a bipartite graph can help us to further understand the FIB encoding width based on tools from graph theory. The next theorem relates the existence of an independent set of a specific size in the bipartite graph G_A to the value of $OPT(A)$.

Theorem 4. *Let A be a two-column FIB table with $|S_1| = |S_2| = n = 2^W$ and let $G_A =< L + R, E >$ be its corresponding bipartite graph. If there does not exist an independent set of vertices $U = U_1 \bigcup U_2$ in G_A, so that $U_1 \subseteq L = S_1$, $U_2 \subseteq L = S_2$ and $|U_1| = |U_2| = \frac{n}{2} + 1$, then the optimal encoding width of A necessarily satisfies $OPT(A) = 2W$. Namely, the optimal encoding width is achieved by the fixed-length encoding and it cannot be improved by any variable-length encoding.*

Proof: Let $C_A = (\sigma_1, \sigma_2)$ be an arbitrary FIB table encoding scheme of A. As mentioned earlier, by Property 2, $OPT(A) \leq 2W$. We would like to show also that $\ell(C_A) \geq 2W$ and therefore $OPT(A) \geq 2W$. By Kraft's inequality, there are (at least) $\frac{n}{2} + 1$ elements in S_1 whose codeword lengths in σ_1 are at least W bits. Let U_1 denote the set of their corresponding vertices in L. Likewise, let $U_2 \subseteq R$ denote the similar set of vertices representing the elements in S_2 with codeword lengths of at least W in σ_2. We then have that $|U_1|, |U_2| \geq \frac{n}{2} + 1$. If $\ell(C_A) < 2W$, then $(\forall i \in [1, h])$, $\ell(\sigma_1(a_1^i)) + \ell(\sigma_2(a_2^i)) \leq 2W - 1$. Since $(\forall x \in U_1, y \in U_2)$, $\ell(\sigma_1(x)) + \ell(\sigma_2(y)) \geq W + W = 2W$, then necessarily, $(\forall x \in U_1, y \in U_2)$, $(x, y) \notin E$. Thus $U = U_1 \bigcup U_2$ is an independent set in G_A. Since $|U_1|, |U_2| \geq \frac{n}{2} + 1$, we have a contradiction, and can deduce that indeed $\ell(C_A) \geq 2W$ and $OPT(A) = 2W$. ∎

B. Optimal Conditional Encoding of the Second Column

We now consider the conditional problem of finding an optimal two-column encoding *given that the encoding of the first column is known*.

Formally, given a two-column FIB table A and a known prefix encoding of one of its columns $\sigma_1 := \bar{\sigma}_1$, we want to find a prefix encoding σ_2 of the second column such that the encoding width $\ell(C_A)$ of the FIB encoding scheme $C_A = (\sigma_1 := \bar{\sigma}_1, \sigma_2)$ is minimized. We denote the FIB encoding width of such scheme by $OPT(A | \sigma_1 := \bar{\sigma}_1)$, i.e.

$$OPT(A | \sigma_1 := \bar{\sigma}_1) = \min_{C_A = (\sigma_1 := \bar{\sigma}_1, \sigma_2)} \ell(C_A). \tag{11}$$

We would like to suggest an algorithm to find such an optimal conditional encoding. Let A be a two-column FIB table with two sets of distinct elements in each of the columns S_1, S_2 and an encoding of the first column $\sigma_1 := \bar{\sigma}_1$. For $y \in S_2$,

we denote by $\phi_{\bar{\sigma}_1}(y)$ the maximal codeword length $\ell(\bar{\sigma}_1(x))$ of an element $x \in S_1$ that shares a row with y in A.

$$\phi_{\bar{\sigma}_1}(y) = \max\{\ell(\bar{\sigma}_1(x)) | x \in S_1, (x,y) \in E\}. \quad (12)$$

By Definition 4 of the FIB encoding width, we can see that the encoding width of an encoding scheme $C_A = (\bar{\sigma}_1, \sigma_2)$ can be presented as

$$\begin{aligned}\ell(C_A) &= \max_{i\in[1,h]} \Big(\ell(\bar{\sigma}_1(a_1^i)) + \ell(\sigma_2(a_2^i))\Big) \\ &= \max_{y\in S_2} \Big(\phi_{\bar{\sigma}_1}(y) + \ell(\sigma_2(y))\Big). \quad (13)\end{aligned}$$

We observe that in order to calculate the optimal encoding σ_2 given that $\sigma_1 := \bar{\sigma}_1$, it is enough to know $\phi_{\bar{\sigma}_1}(y)$ for any $y \in S_2$ while the exact values of $\ell(\bar{\sigma}_1(s_{1,1})), \ldots, \ell(\bar{\sigma}_1(s_{1,n}))$ are not necessarily required. This is because

$$\begin{aligned}OPT(A|\sigma_1 := \bar{\sigma}_1) &= \min_{C_A=(\sigma_1:=\bar{\sigma}_1,\sigma_2)} \ell(C_A) \\ &= \min_{C_A=(\sigma_1:=\bar{\sigma}_1,\sigma_2)} \max_{i\in[1,h]} \Big(\ell(\bar{\sigma}_1(a_1^i)) + \ell(\sigma_2(a_2^i))\Big) \\ &= \min_{C_A=(\sigma_1:=\bar{\sigma}_1,\sigma_2)} \max_{y\in S_2} \Big(\phi_{\bar{\sigma}_1}(y) + \ell(\sigma_2(y))\Big). \quad (14)\end{aligned}$$

Before presenting the algorithm, we suggest the following lemma.

Lemma 5. *There exists a (prefix) encoding $\bar{\sigma}_2$ of S_2 such that $(\sigma_1, \sigma_2) = (\bar{\sigma}_1, \bar{\sigma}_2)$ achieves the minimal encoding width of all the FIB encoding schemes $\{(\sigma_1, \sigma_2) | \sigma_1 := \bar{\sigma}_1\}$ and satisfies*

$$(\forall y \in S_2), \phi_{\bar{\sigma}_1}(y) + \ell(\bar{\sigma}_2(y)) = OPT(A|\sigma_1 := \bar{\sigma}_1). \quad (15)$$

Proof: Let $\widehat{C}_A = (\sigma_1 := \bar{\sigma}_1, \widehat{\sigma}_2)$ be a FIB encoding scheme that satisfies $\ell(\widehat{C}_A) = OPT(A|\sigma_1 := \bar{\sigma}_1)$. If $(\forall y \in S_2), \phi_{\bar{\sigma}_1}(y) + \ell(\widehat{\sigma}_2(y)) = OPT(A|\sigma_1 := \bar{\sigma}_1)$ then $\bar{\sigma}_2 = \widehat{\sigma}_2$ can be the requested encoding. Otherwise, by the definition of $\ell(\widehat{C}_A)$, $(\forall y \in S_2), \phi_{\bar{\sigma}_1}(y) + \ell(\widehat{\sigma}_2(y)) \le \ell(\widehat{C}_A) = OPT(A|\sigma_1 := \bar{\sigma}_1)$. This is because by Equation (13), for $\widehat{C}_A = (\bar{\sigma}_1, \widehat{\sigma}_2)$ we have $\ell(\widehat{C}_A) = \max_{y\in S_2} \Big(\phi_{\bar{\sigma}_1}(y) + \ell(\widehat{\sigma}_2(y))\Big)$. For $y \in S_2$ we use the notation $q(y) = OPT(A|\sigma_1 := \bar{\sigma}_1) - (\phi_{\bar{\sigma}_1}(y) + \ell(\widehat{\sigma}_2(y)))$. We define an encoding $\bar{\sigma}_2$ of S_2 based on $\widehat{\sigma}_2$ as $\bar{\sigma}_2(y) = \widehat{\sigma}_2(y) \cdot 0^{q(y)}$ where the $\cdot$ sign is used to represent the concatenation operation. In particular, if for $y \in S_2$, $\phi_{\bar{\sigma}_1}(y) + \ell(\widehat{\sigma}_2(y)) = OPT(A|\sigma_1 := \bar{\sigma}_1)$ then $\bar{\sigma}_2(y) = \widehat{\sigma}_2(y) \cdot 0^{q(y)} = \widehat{\sigma}_2(y)$. We can see that $(\forall y \in S_2), \phi_{\bar{\sigma}_1}(y) + \ell(\bar{\sigma}_2(y)) = \phi_{\bar{\sigma}_1}(y) + \ell(\widehat{\sigma}_2(y)) + q(y) = OPT(A|\sigma_1 := \bar{\sigma}_1)$. Thus $\bar{\sigma}_2$ satisfies the required equality.

To complete the proof we have to show also that $\bar{\sigma}_2$ is indeed a prefix encoding. We can first verify that the $n_2 = |S_2|$ codewords of σ_2 are different. If for any $x, y \in S_2$ (s.t. $x \neq y$), $\bar{\sigma}_2(x) = \widehat{\sigma}_2(x) \cdot 0^{q(x)} = \widehat{\sigma}_2(y) \cdot 0^{q(y)} = \bar{\sigma}_2(y)$ then necessarily $\widehat{\sigma}_2(x)$ is a prefix of $\widehat{\sigma}_2(y)$ (if $q(x) \ge q(y)$) or $\widehat{\sigma}_2(y)$ is a prefix of $\widehat{\sigma}_2(x)$ (if $q(x) \le q(y)$), a contradiction since $\widehat{\sigma}_2$ is a prefix encoding. $\bar{\sigma}_2$ also preservers the prefix property since if $\bar{\sigma}_2(x) = \widehat{\sigma}_2(x) \cdot 0^{q(x)}$ is a prefix of $\bar{\sigma}_2(y) = \widehat{\sigma}_2(y) \cdot 0^{q(y)}$ then again either $\widehat{\sigma}_2(x)$ is a prefix of $\widehat{\sigma}_2(y)$ or $\widehat{\sigma}_2(y)$ is a prefix of $\widehat{\sigma}_2(x)$. ∎

Based on the last lemma we can deduce from Kraft's inequality, the exact value of $OPT(A|\sigma_1 := \bar{\sigma}_1)$.

Theorem 6. *The optimal conditional FIB encoding width of A satisfies*

$$OPT(A|\sigma_1 := \bar{\sigma}_1) = \left\lceil \log_2 \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil. \quad (16)$$

Proof: We first would like to show that $OPT(A|\sigma_1 := \bar{\sigma}_1) \ge \left\lceil \log_2 \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil$. Consider a FIB encoding $(\sigma_1, \sigma_2) = (\bar{\sigma}_1, \bar{\sigma}_2)$ that has an encoding width of $OPT(A|\sigma_1 := \bar{\sigma}_1)$ and $(\forall y \in S_2)$ satisfies $\phi_{\bar{\sigma}_1}(y) + \ell(\bar{\sigma}_2(y)) = OPT(A|\sigma_1 := \bar{\sigma}_1)$. The last equality can be presented also as $\ell(\bar{\sigma}_2(y)) = OPT(A|\sigma_1 := \bar{\sigma}_1) - \phi_{\bar{\sigma}_1}(y)$. Such an encoding exists by Lemma 5. By Kraft's inequality $\bar{\sigma}_2$ satisfies

$$\begin{aligned}1 \ge \sum_{y\in S_2} 2^{-\ell(\bar{\sigma}_2(y))} &= \sum_{y\in S_2} 2^{-\Big(OPT(A|\sigma_1:=\bar{\sigma}_1) - \phi_{\bar{\sigma}_1}(y)\Big)} \\ 2^{OPT(A|\sigma_1:=\bar{\sigma}_1)} &\ge \sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \\ OPT(A|\sigma_1 := \bar{\sigma}_1) &\ge \left\lceil \log_2 \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil, \quad (17)\end{aligned}$$

where the last inequality is obtained since $OPT(A|\sigma_1 := \bar{\sigma}_1)$ is of course an integer. To complete the proof we would also like to show that $OPT(A|\sigma_1 := \bar{\sigma}_1) \le \left\lceil \log_2 \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil$ by finding a FIB encoding scheme $(\sigma_1, \sigma_2) = (\bar{\sigma}_1, \bar{\sigma}_2)$ whose encoding width satisfies this upper bound. To do so, we denote by $v_{\bar{\sigma}_1}$ the term $\left\lceil \log_2 \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil$. Then,

$$\begin{aligned}\sum_{y\in S_2} 2^{-(v_{\bar{\sigma}_1} - \phi_{\bar{\sigma}_1}(y))} &= \sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y) - v_{\bar{\sigma}_1}} = \sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \cdot 2^{-v_{\bar{\sigma}_1}} \\ &\le \sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \cdot \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big)^{-1} = 1. \quad (18)\end{aligned}$$

Thus there exists a prefix encoding $\bar{\sigma}_2$ with codeword lengths $\ell(\bar{\sigma}_2(y)) = (v_{\bar{\sigma}_1} - \phi_{\bar{\sigma}_1}(y))$ for $y \in S_2$. The FIB encoding scheme $C_A = (\bar{\sigma}_1, \bar{\sigma}_2)$ satisfies

$$\begin{aligned}\ell(C_A) &= \max_{y\in S_2} \Big(\phi_{\bar{\sigma}_1}(y) + \ell(\bar{\sigma}_2(y))\Big) \quad (19) \\ &= \max_{y\in S_2} \Big(\phi_{\bar{\sigma}_1}(y) + (v_{\bar{\sigma}_1} - \phi_{\bar{\sigma}_1}(y))\Big) \\ &= \max_{y\in S_2} \Big(v_{\bar{\sigma}_1}\Big) = v_{\bar{\sigma}_1}.\end{aligned}$$

Then, $OPT(A|\sigma_1 := \bar{\sigma}_1) \le v_{\bar{\sigma}_1} = \left\lceil \log_2 \Big(\sum_{y\in S_2} 2^{\phi_{\bar{\sigma}_1}(y)} \Big) \right\rceil$. ∎

Now, when the codeword lengths of the elements of S_2 are known from Lemma 5 and Theorem 6, it is easy to find such an encoding as explained earlier in Section III.

Based on the suggested optimal conditional encoding from above, we now suggest two directions to find an efficient encoding scheme for a given two-column FIB table. Of course, these directions are alternatives for the encoding of general FIB tables presented in Section III-B that can be applied also

for two-column tables. The first direction may be considered when the available running time is very limited while the second direction requires a longer running time.

(i) We first count for each of the elements in the first column, the number of distinct entries in which it appears. As discussed earlier, duplicated entries have no influence on the obtained encoding width. Based on these numbers of distinct appearances, we can calculate a corresponding distribution. We can then encode this first column by a Huffman coding [25] based on this distribution. Given this encoding of the first column, we can find an encoding scheme for the whole table by encoding the second column by the optimal conditional encoding given the encoding of the first column. Another option for an encoding scheme is obtained similarly by considering the two columns in a reverse order. By Lemma 5 and Theorem 6, it is guaranteed that both options perform at least as well as an encoding scheme composed of the two Huffman codings for each of the two columns. Finally, we can encode the given two-column FIB tables by the best of the two options, i.e. the one that yields a smaller encoding width.

(ii) We find the encoding scheme obtained by the algorithm for general FIB tables presented in Section III-B. Then, we consider (independently) as the given (constrained) encoding, each of the two encodings for the two columns composing the solution. For each of them, we find the optimal conditional encoding of the other column. Again, by Lemma 5 and Theorem 6, it is guaranteed that both options perform at least as well as the original solution given by the algorithm.

V. Lower Bounds on the Optimal Encoding Width of Two-Column FIB Tables

We would like now to suggest additional lower bounds on the optimal encoding width of a given two-column FIB table A as defined above. We can calculate this bound without solving the relaxed FIB encoding problem.

Let again $G_A =< L + R, E >=< S_1 + S_2, E >$ with $|S_1| = |S_2| = n = 2^W$. We first recall an additional definition from graph theory.

Definition 5 (Edge Cover)**.** *For an undirected graph $G =< V, E >$, a set of edges $S \subseteq E$ is called an* edge cover *of G if $\bigcup_{(x,y)\in S}\{x, y\} = V$, i.e. each of the vertices in V is incident on at least one of the edges in S.*

Clearly, G_A does not include any isolated vertices, i.e. any vertex is connected to at least one vertex and the value it represents appears in at least one of the rows of A. Therefore, an edge cover always exists. For a graph G, we denote by $\rho(G)$ the edge covering number of G, i.e. the minimal number of edges in an edge cover of G. We first show a simple property of $\rho(G_A)$.

Lemma 7. *The edge covering number $\rho(G_A)$ of G_A satisfies $n \leq \rho(G_A) \leq 2 \cdot (n-1)$.*

Proof: Consider the edges in a minimal edge cover in an arbitrary order. The first edge covers two vertices. Any additional edge adds one or two vertices to the set of vertices covered by previous edges in the edge cover. Since there are $2n$ vertices in G_A then clearly $n \leq \rho(G_A) \leq 2n-1$. To show that $\rho(G_A) \leq 2n-2$, we show that there must be two independent edges in G_A, i.e. edges that do not share any vertex. Consider an edge $(s_{1,i}, s_{2,j})$ for $i, j \in [1, n]$. If all the other edges in E share at least one vertex with this edge, we must have that $E = \{(s_{1,i}, s_{2,k}) | k \in [1, n]\} \bigcup \{(s_{1,k}, s_{2,j}) | k \in [1, n]\}$. Then, for instance the edges $(s_{1,i}, s_{2,k}), (s_{1,k}, s_{2,j})$ for $k \neq i, j$ are independent. Finally, the two independent edges can contribute two covered vertices to the edge cover and necessarily $\rho(G_A) \leq 2n - 2 = 2 \cdot (n-1)$. ∎

Let $\alpha = \rho(G_A)/n$. The next theorem suggests a lower bound on the optimal encoding width of A based on the value of $\rho(G_A)$.

Theorem 8. *The optimal encoding width of A satisfies*

$$OPT(A) \geq \left\lceil \frac{2 \cdot (W + \alpha - 1)}{\alpha} \right\rceil. \tag{20}$$

Proof: For $G_A =< S_1 + S_2, E >$, let $S \subseteq E$ be an edge cover of G_A of size $\rho(G_A) = \alpha \cdot n$. Let $C_A = (\sigma_1, \sigma_2)$ be an arbitrary FIB table encoding scheme of A. We will show that

$$\max_{(x,y)\in S} \ell(\sigma_1(x)) + \ell(\sigma_2(y)) \geq \left\lceil \frac{2 \cdot (W + \alpha - 1)}{\alpha} \right\rceil. \tag{21}$$

Thus, since $S \subseteq E$ then necessarily

$$\ell(C_A) = \max_{(x,y)\in E} \ell(\sigma_1(x)) + \ell(\sigma_2(y)) \geq \left\lceil \frac{2 \cdot (W + \alpha - 1)}{\alpha} \right\rceil. \tag{22}$$

To do so, we consider the subset of the rows of A represented by the edge cover S. Let $D = ((d_1^1, d_2^1), \ldots, (d_1^{\rho(G_A)}, d_2^{\rho(G_A)}))$ represent these rows, s.t. $|D| = \rho(G_A) = \alpha \cdot n$. By Definition 5 of the edge cover, $S_1 = \bigcup_{i=1}^{\rho(G_A)} \{d_1^i\}$ and $S_2 = \bigcup_{i=1}^{\rho(G_A)} \{d_2^i\}$, i.e. all the elements of S_1 appear at least once in the first column of the rows of D as well as all the elements of S_2 in the second. Let $X_D(C_A)$ denote the total sum of codeword lengths of the elements in D using the encoding $C_A = (\sigma_1, \sigma_2)$, s.t. $X_D(C_A) = \sum_{i=1}^{\rho(G_A)} \Big(\ell(\sigma_1(d_1^i)) + \ell(\sigma_2(d_2^i))\Big)$.

By manipulating Kraft's inequality, we can have that the total sum of codeword lengths of the n distinct elements in each column is at least $n \cdot W$. Of course, the other $\alpha \cdot n - n = (\alpha - 1) \cdot n$ elements in each column are encoded in at least one bit each. Thus,

$$\begin{aligned} X_D(C_A) &= \sum_{i=1}^{\rho(G_A)} \Big(\ell(\sigma_1(d_1^i)) + \ell(\sigma_2(d_2^i))\Big) \\ &= \sum_{i=1}^{\rho(G_A)} \ell(\sigma_1(d_1^i)) + \sum_{i=1}^{\rho(G_A)} \ell(\sigma_2(d_2^i)) \\ &\geq n \cdot W + (\alpha - 1) \cdot n + n \cdot W + (\alpha - 1) \cdot n \\ &= 2n \cdot W + 2(\alpha - 1) \cdot n. \end{aligned} \tag{23}$$

We recall that the total number of rows is $\alpha \cdot n$ and thus there is at least one row in D that is encoded in at least $\left\lceil \frac{X_D(C_A)}{\alpha \cdot n} \right\rceil$

(a) (b)

Fig. 3. Two examples of bipartite graphs with $n = 2^W = 8$ (for $W = 3$) vertices in each graph side representing two two-column FIB tables A_1 and A_2. In the first graph (a), there is a perfect match and the size of the (minimal) edge cover is $8 = n = \alpha \cdot n$ with $\alpha = 1$ and $OPT(A_1) = 2W = 6$. In the second (b), the size of the edge cover is $14 = \alpha \cdot n$ with $\alpha = 1.75$ and $OPT(A_2) \geq 5$.

bits. Thus the encoding width of C_A satisfies

$$\begin{aligned}\ell(C_A) &= \max_{(x,y)\in E} \ell(\sigma_1(x)) + \ell(\sigma_2(y)) \\ &\geq \max_{(x,y)\in S} \ell(\sigma_1(x)) + \ell(\sigma_2(y)) \geq \left\lceil \frac{X_D(C_A)}{\alpha \cdot n} \right\rceil \\ &\geq \left\lceil \frac{2n \cdot W + 2(\alpha-1) \cdot n}{\alpha \cdot n} \right\rceil \\ &= \left\lceil \frac{2 \cdot (W + \alpha - 1)}{\alpha} \right\rceil \geq \left\lceil \frac{2W}{\alpha} \right\rceil . \end{aligned} \tag{24}$$

Finally, Since C_A is an arbitrary encoding scheme, we can deduce the result. ■

Example 4. *Figure 3 illustrates the two corresponding bipartite graphs for two-column FIB tables A_1 and A_2, with $n = 2^W = 8$ (for $W = 3$) distinct elements in each column.*

For A_1, the bipartite graph includes a perfect match and the size of the edge cover is $8 = n = \alpha \cdot n$ with $\alpha = 1$. Theorem 8 implies that $OPT(A_1) \geq \left\lceil \frac{2\cdot(W+\alpha-1)}{\alpha} \right\rceil = 2W$.

For A_2, the size of the minimal edge cover is $14 = 2 \cdot (n - 1) = \alpha \cdot n$ with $\alpha = 1.75$. Thus $OPT(A_2) \geq \left\lceil \frac{2\cdot(3+1.75-1)}{1.75} \right\rceil = 5$. Further, by encoding the first element in a column in a single bit and others in four bits, we have that indeed $OPT(A_2) = 5$.

VI. Supporting Updates

Forwarding tables need to support updates. These updates can include an insertion of a new entry, a deletion of an existing entry, or a modification of some of the fields of an entry. For instance, they may follow the addition or deletion of VMs in the datacenter, as well VM migrations. In this section, we discuss how such updates can be supported.

While updating the forwarding table in its encoded representation, we might encounter several obstacles. First, an entry might have to be encoded in a width larger than the currently allocated memory width. In addition, in some cases there may be a problem to allocate additional codewords for new inserted elements while still keeping the properties of the prefix encodings.

In general, the update process includes two steps: *an immediate step required for coherency, and another procedure that can occur offline to improve performance*. This second step can be run, for instance, either periodically in time or after a fixed number of updates. Meanwhile, in addition to the encoded table, we make use of a small dedicated memory for keeping a small number of entries. Let P be the fixed allocated number of bits for the encoding of each entry. To support such updates, we assume that each entry includes an additional bit that indicates its validity.

Insertion and Modification: Dealing with with an insertion of a new entry or a change in an entry can be done in a similar way. We describe several possible scenarios. First, *if all the elements in the updated entry appear in the current dictionaries*, we simply encode a new entry based on these dictionaries. If its total width is not greater than P, no further changes are required. If it is greater than P, we set the original entry (in case of a change) as invalid and temporarily add the new entry (without encoding it) to the small memory.

If the updated entry includes a new element, we try to associate it with a codeword. In a specific column, we can do so only if the sum that appears in Kraft's inequality is smaller than one. Further, the minimal possible codeword length depends on the value of this sum. For the sake of simplicity, we suggest to allocate for each new element a codeword with the minimal possible length. If we cannot allocate any new codeword in one of these columns (due to an equality in the corresponding Kraft's inequality), or if the allocated codewords yield an entry that is too long, the updated entry is again added to the small memory.

Deletion: Dealing with *an entry deletion* is easy and requires setting the entry as invalid. Such a change does not require any changes in the encodings of other entries, since the maximal width of the rest of the encoded entries is clearly bounded by the maximal width before change. Reduction in the maximal width of the table after an entry removal may be achieved by running the compression algorithm as part of the offline procedure.

Offline Process: The offline process includes the calculation of an efficient encoding for the existing encoded entries and for the unencoded entries stored in the small memory. Later, this memory is cleared. The value of P may be changed, as well as the encoded version of existing entries. It is possible to endow the table encoding algorithm with better update capabilities by choosing an encoding with a slightly larger value of P that guarantees short codeword lengths for possible future inserted elements. Another improvement can be as follows. To avoid the relatively long running time of the process, we can try to rely on the previously found encoding scheme. We consider the elements in the problematic entries stored in the additional memory. We try to allocate for these elements relatively short codewords, to enable them to be within the allocated memory width. To do so, we might have to replace by longer codewords the representation of other elements that appear in shorter entries. If we do not succeed to do so, we can repeat the process after increasing the current memory width. Meanwhile, for coherency reasons, between two offline processes, any lookup should consider the encoded entries in the FIB and the unencoded entries in the small dedicated memory. In addition, there might be a synchronization problem to deal with access requests during the short time required for changing the memory (e.g. in the completion of the offline process). We suggest to postpone these access requests for a short time. If this is impossible, we can keep an additional copy of the table that will be accessed

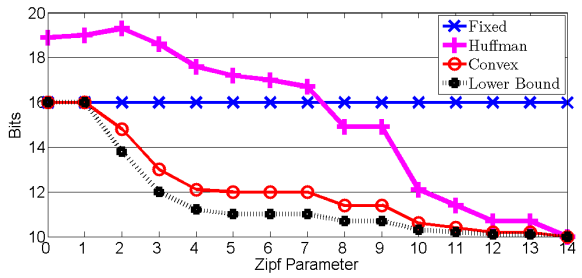
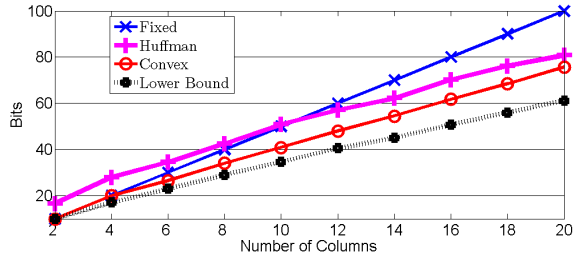
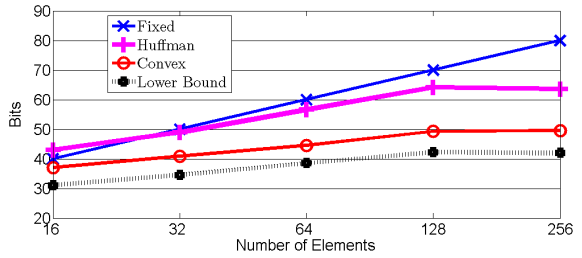


Fig. 4. Effectiveness of the encoding algorithm on synthetic FIB tables with $d = 2$ columns and $W = 8$, as a function of the Zipf parameter. Elements are selected according to the Zipf distribution with the given parameter. By Theorem 3, the optimal encoding width is at least $W + 2 = 10$ and at most $2W = 16$. Since $d = 2$, our suggested *convex* scheme takes at most $d - 1 = 1$ bit more than the theoretical lower bound. In comparison, the fixed-length encoding has a fixed width of $2W = 16$ bits, while the Huffman encoding sometimes achieves a greater width.



(a) Effectiveness of the encoding algorithm as a function of the number of columns with $2^W = 32$ distinct elements per column.



(b) Effectiveness of the encoding algorithm as a function of the number of distinct elements per column with $d = 10$ columns.

Fig. 5. Analysis of the encoding algorithm as a function of table parameters.

in such cases.

VII. Experimental Results

We now turn to conducting experiments to examine the distribution of the optimal encoding width and the performance of our suggested encoding algorithm from Section III-B. In the experiments, we rely on both synthetic and real-life FIB tables.

A. Effectiveness on Synthetic FIB tables

We first evaluate our suggested scheme on *synthetic two-column FIB tables*. In our first experiments, each table has 1000 entries with $2^W = 256$ (for $W = 8$) distinct elements in each of the $d = 2$ columns. To generate the table, we first make sure that each element appears at least once in each column by adding an entry containing it with an additional random element in the other column. This additional random element is selected according to a Zipf distribution of parameter s. Then, we add additional entries with $d = 2$ random elements selected according to the same distribution. Intuitively, a Zipf distribution with a lower parameter s is closer to the uniform distribution.

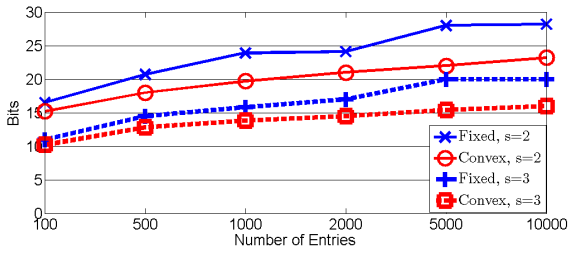


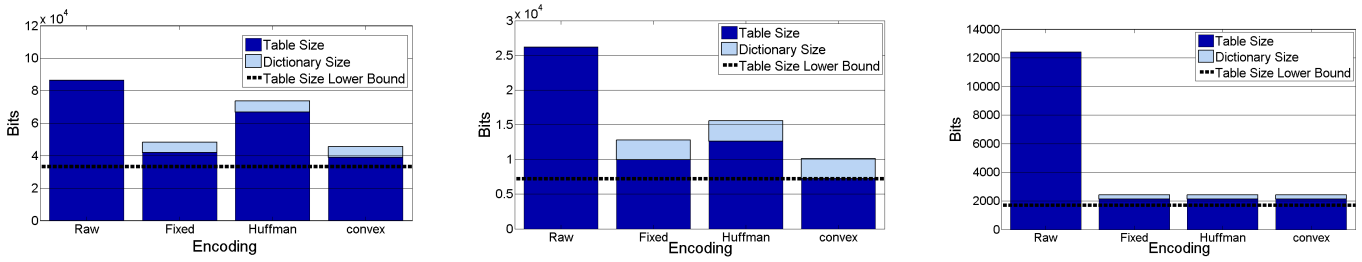
Fig. 6. Effectiveness of the number of entries on the encoding width in synthetic FIB tables with $d = 4$ columns and at most $2^W = 256$ (for $W = 8$) distinct elements in each column. Two values of the Zipf parameter $s = 2, 3$ are considered.

First, in Fig. 4, we compare our suggested scheme (from Section III-B), denoted "convex", with the fixed-length and Huffman encoding schemes. The results are based on the average of 10 simulations. We present the lower bound given by the fractional solution of the relaxed optimization problem (Section III-B). We also show our suggested scheme, which takes the ceiling of the fractional solution of the relaxed problem, unless it exceeds $2W$, in which case it simply uses fixed-length encoding. As suggested by Theorem 1, our scheme is always within $d - 1 = 1$ bit of the fractional-optimum lower bound. For instance, for a Zipf parameter $s = 4$, the widths for the Huffman, fixed-length, and our suggested encodings are respectively 17.6, $2W = 16$, and 12.1, while the lower bound is 11.2. More generally, the lower the Zipf parameter, the closer the lower bound is to $2W$.

Next, in Fig. 5, we plot the encoding efficiency as a function of the number of columns (a) and of the number of elements appearing in each column (b). In both plots we take $h = 1000$ entries and a Zipf parameter $s = 2$. In Fig. 5(a) the number of elements is 32, i.e. $W = 5$, and in Fig. 5(b) the number of columns is $d = 10$. We can see that in (a), Huffman becomes better for the worst-case entry width. Intuitively, it is because the ratio of the standard deviation of the sum of the encodings of d columns to the expected sum decreases with d as $\frac{1}{\sqrt{d}}$.

We want to stress that the presented encoding scheme suggests a tradeoff between memory requirements and running time. While our scheme performs better and obtains an improved encoding width, it is also much slower to run than the simpler schemes. As mentioned in the implementation section, it is designed for offline or background use. For instance, in Fig. 5(a), an unoptimized implementation of our scheme on a basic Intel i5 core respectively took 0.82 seconds and 23.1 seconds for the two-column and ten-column tables, while the Huffmann scheme ran in the respectively- negligible times of 7 and 11 milliseconds.

We perform an additional simulation to examine how the number of entries influence the encoding width. We compare again our suggested scheme with the fixed-



(a) United States (b) China 1 (c) China 2

Fig. 7. Total memory size of encoded real-life tables including the dictionary size. All forwarding tables were collected in enterprise networks, in the United States as well as in China.

length encoding in tables of h entries for $h \in \{100, 500, 1000, 2000, 5000, 10000\}$ and $d = 4$ columns. All entries are selected randomly among $2^W = 256$ (for $W = 8$) possible elements in each column, according to Zipf distribution parameters $s = 2, 3$. Here, since we consider also tables with relatively small number of entries, we do not force all elements to appear at least once in each column. Accordingly, the fixed-length encoding yields an encoding width smaller than $d \cdot W = 32$. The results are based on the average of 10 simulations.

Fig. 6 depicts the results. As expected, the encoding width of our scheme increases when the number of entries grows since the additional entries result in more constraints on the width. For all examined number of entries the improvement of our scheme is not negligible. For instance, if $h = 500$ and $s = 3$ the obtained width is 12.8 in comparison with 14.5 for the fixed-length encoding (an improvement of 11.7%). Likewise, if $h = 2000$ the width is 14.5 instead of 17.0 (14.9% improvement). The improvement is even larger for the case of $h = 10000$ entries and equals 20.0%. Further, we expect to observe an improvement for tables with relatively large number of entries as long as the distribution of their elements is less close to the uniform distribution. Of course, with infinite number of entries, all possible entries appear and we cannot improve the fixed-length encoding.

B. Real-Life FIB tables

We also conduct experiments on three typical real-life enterprise-network tables. The tables are collected from three different switches of three different switch vendors, each switch using a different number of columns. All tables were collected in enterprise networks, the first in the United States with 2790 entries and the other in China with 903 and 428 entries.

We would first like to evaluate the various encoding schemes. For each table, we present the size of the original raw table without compression. We compare it to the total size of the compressed table, including the dictionary for three compression algorithms: the fixed-length encoding, the Huffman encoding and our suggested encoding. The results are presented in Fig. 7.

For instance, in the second real-life table (China 1), presented in Fig. 7 (b), we consider the 903 entries of two columns representing the VLAN and the Target-Port fields.

	number of fields	number of entries	raw data width	fixed-length width	improved width	long entries	uncoded entries
United States	3	2790	31	15	13	53	0
China 1	2	903	29	11	8	13	2
China 2	2	428	29	5	5	3	0

TABLE II
DEALING WITH RULE UPDATES. ENCODING IS BASED ON THE FIRST HALF OF THE ENTRIES IN EACH FILE. ADDITIONAL ENTRIES ARE CONSIDERED AS NEW INSERTED ENTRIES.

Without encoding, each entry is allocated 29 bits per entry and thus the raw data without encoding requires $903 \cdot 29 = 26187$ bits. Using the three encodings, the size of the dictionaries is almost fixed and equals approximately 2900 bits. An entry width of 11 and 14 bits is achieved in the fixed-length encoding and in the Huffman encoding, respectively, while in the proposed encoding we achieve an entry width of only 8 bits. This leads to an improvement of 20.7%, 34.9%, and 61.3% in the total memory size (including the dictionary) compared to the fixed-length, Huffman and Raw solutions, respectively.

We would also like to examine the efficiency of the encoding scheme while dealing with forwarding table updates. To illustrate updates we perform the following experiment. For each of the three real-life tables, we consider the first half of entries (e.g. 1395 out of 2790 in the table from United States). We then find for each of them, our suggested encoding scheme. Next, we consider the entries in the second half of the original table as new inserted entries. We examine the ability to encode the new entries within the determined encoding width. Another obstacle is that these new entries might include new elements that are not encoded in the encoding scheme found for the first half of entries. Our high interest in insertions of new elements is explained by the fact that, as mentioned in Section VI, changes in existing entries can be also supported by insertions of new entries.

The results are summarized in Table II. In general, only a small portion of the new entries could not be encoded within the predetermined width. For instance, in the table from United States an encoding width of 13 bits was obtained in the encoding of the first 1395 entries, improving the width of 15 bits required for the encoding of the file with the fixed-length encoding. While trying to insert the last 1395 entries of the second half of the file as new entries, 1342 of them (96.2%) could be encoded successfully within 13 bits based

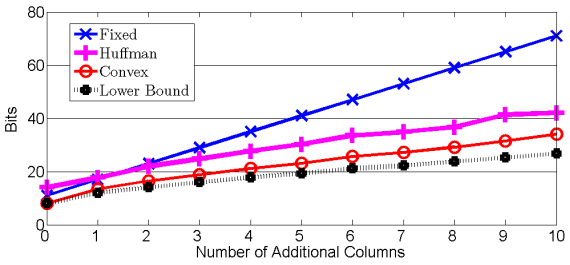


Fig. 8. Effectiveness of the encoding algorithm on the extended table (based on the China 1 table), as a function of the number of columns.

on the previously selected encoding scheme. Only 53 entries obtained widths larger than 13. In addition, none of these new entries contained new elements such that their encoding was not defined in the scheme for the first entries. Likewise, in China 1, we encountered 13 entries with overflow among 452 inserted entries (only 2.88%). Only two of the new entries contained elements that did not appear in the first half of entries.

C. Extrapolated Tables

We want to extrapolate the behaviors of these schemes from real-life limited data sets to the larger expected data sets to be encountered in future datacenter network architectures. The approach we take herein is to amplify the available data sets by scaling out the number of columns in the table, so as to model a table with a scaled-out number of attributes.

We extend the China 1 table by duplicating and appending the port column several times. In each new column, we permute the appearances of the elements uniformly at random. The motivation is to model additional forwarding attributes that would follow a distribution that is similar to an existing one. Rearranging the row locations of the duplicate elements adds randomness to the extended file. As seen in Fig. 8, for each additional column the fixed encoding increases by a constant, as expected since the number of unique elements in each column does not vary. The Huffman and convex encodings increase at a slower pace, thereby increasing the encoding effectiveness as the number of columns grows. Of course, since this table is extended synthetically, we urge caution in over-interpreting the results.

VIII. Conclusion

In this paper, we saw how datacenter virtualization is causing a dramatic rise in the number of entries in forwarding tables, such that it becomes impossible to implement forwarding tables in current memory sizes without any compression algorithm. We then investigated the compressibility of forwarding tables, by introducing a novel forwarding table architecture with separate encoding in each column. Our architecture supports fast random accesses and fixed-width memory words. We also later explained how our architecture can handle table updates.

Later, we suggested an encoding whose per-entry memory requirement is guaranteed to be within a small additive constant of the optimum. Finally, we evaluated this new scheme against the fixed-width and Huffman schemes, using both synthetic and real-life forwarding tables from different switches, switch vendors, and network country locations. As future work, we plan to check how using additional hardware such as CAM and TCAM can help in further compressing the forwarding tables.

IX. Acknowledgment

We would like to thank Yishay Mansour and Haim Kaplan for their helpful participation and suggestions.

This work was partly supported by the Andrew Viterbi graduate fellowship, by the Google European Doctoral fellowship, by a European Commission Marie Curie CIG grant, by the Intel ICRI-CI Center, by the Israel Science Foundation grant No. 1241/12, by the German-Israeli Foundation for Scientific Research and Development, by the Japan Technion Society and Greenberg (Ottawa) Research Funds, and by the Israel Ministry of Science and Technology.

References

[1] O. Rottenstreich, M. Radan, Y. Cassuto, I. Keslassy, C. Arad, T. Mizrahi, Y. Revah, and A. Hassidim, "Compressing forwarding tables," in *IEEE INFOCOM*, 2013.

[2] I. Gashinsky, "Datacenter scalability panel," in *North American Network Operators Group, NANOG 52*, 2011.

[3] R. N. Mysore *et al.*, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *SIGCOMM*, 2009.

[4] T. Narten *et al.*, "Problem statement for ARMD," IETF, draft-ietf-armd-problem-statement, work in progress, 2012.

[5] G. Hankins, "Pushing the limits, a perspective on router architecture challenges," in *North American Network Operators Group, NANOG 53*, 2011.

[6] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," IETF, RFC 4984, 2007.

[7] T. M. Cover and J. A. Thomas, *Elements of information theory (2. ed.)*. Wiley, 2006.

[8] T. Narten, "On the scalability of internet routing," IETF, draft-narten-radir-problem-statement, work in progress, 2010.

[9] P. Francis, X. Xu, H. Ballani, D. Jen, R. Raszuk, and L. Zhang, "FIB suppression with virtual aggregation," IETF, draft-ietf-grow-va, work in progress, 2011.

[10] X. Zhao, Y. Liu, L. Wang, and B. Zhang, "On the aggregatability of router forwarding tables," in *INFOCOM*, 2010.

[11] R. Draves, C. King, V. Srinivasan, and B. Zill, "Constructing optimal IP routing tables," in *INFOCOM*, 1999.

[12] Q. Li, D. Wang, M. Xu, and J. Yang, "On the scalability of router forwarding tables: Nexthop-selectable FIB aggregation," in *Infocom Mini-Conference*, 2011.

[13] Y. Liu, X. Zhao, K. Nam, L. Wang, and B. Zhang, "Incremental forwarding table aggregation," in *GLOBECOM*, 2010.

[14] Z. A. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis, "Smalta: practical and near-optimal FIB aggregation," in *CoNEXT*, 2011.

[15] M. Pöss and D. Potapov, "Data compression in Oracle," in *VLDB*, 2003.

[16] R. Johnson, V. Raman, R. Sidle, and G. Swart, "Row-wise parallel predicate evaluation," *PVLDB*, vol. 1, no. 1, pp. 622–634, 2008.

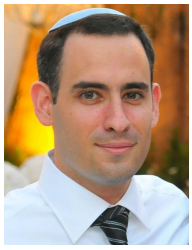
[17] K. Stolze, V. Raman, R. Sidle, and O. Draese, "Bringing BLINK closer to the full power of SQL," in *BTW*, 2009.

[18] O. Rottenstreich, A. Berman, Y. Cassuto, and I. Keslassy, "Compression for fixed-width memories," in *IEEE ISIT*, 2013.

[19] C. R. Meiners, A. X. Liu, and E. Torng, "Topological transformation approaches to TCAM-based packet classification," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 237–250, 2011.

[20] O. Rottenstreich, R. Cohen, D. Raz, and I. Keslassy, "Exact worst-case TCAM rule expansion," *IEEE Trans. Computers*, vol. 62, no. 6, pp. 1127–1140, 2013.

[21] O. Rottenstreich, I. Keslassy, A. Hassidim, H. Kaplan, and E. Porat, "On finding an optimal TCAM encoding scheme for packet classification," in *IEEE Infocom*, 2013.
[22] Y. Kanizo, D. Hay, and I. Keslassy, "Optimal fast hashing," in *IEEE Infocom*, 2009.
[23] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. Society for Industrial Mathematics, 1987, vol. 13.
[24] O. Rottenstreich *et al.*, "Compressing forwarding tables," Technion, Israel, Technical Report TR12-03, 2012. [Online]. Available: http://webee.technion.ac.il/~isaac/papers.html
[25] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, no. 9, 1952.



Ori Rottenstreich received the B.S. degree in computer engineering (summa cum laude) from the electrical engineering department of the Technion, Haifa, Israel in 2008. He is now pursuing a Ph.D. degree in the same department. He is mainly interested in computer networks and especially in exploring novel coding techniques for networking applications. Ori Rottenstreich is a recipient of the Google Europe Fellowship in Computer Networking, the Andrew Viterbi graduate fellowship, the Jacobs-Qualcomm fellowship, the Intel graduate fellowship and the Gutwirth Memorial fellowship. He is a co-recipient of the Best Paper Runner Up Award at the IEEE Infocom 2013 conference.

Marat Radan is an M.S. student at the department of Electrical Engineering of the Technion, Israel. He received his B.S. degree in Electrical engineering from the same department in 2011. His primary research interest is the design and analysis of high-performance routers.

Yuval Cassuto (S'02, M'08) is a faculty member at the Department of Electrical Engineering, Technion - Israel Institute of Technology. His research interests lie at the intersection of the theoretical information sciences and the engineering of practical computing and storage systems. During 2010-2011 he has been a Scientist at EPFL, the Swiss Federal Institute of Technology in Lausanne. From 2008 to 2010 he was a Research Staff Member at Hitachi Global Storage Technologies, San Jose Research Center. He received the B.Sc degree in Electrical Engineering, summa cum laude, from the Technion in 2001, and the M.S. and Ph.D. degrees in Electrical Engineering from the California Institute of Technology, in 2004 and 2008, respectively. From 2000 to 2002, he was with Qualcomm, Israel R&D Center, where he worked on modeling, design and analysis in wireless communications. Dr. Cassuto has won the 2010 Best Student Journal Paper Award in data storage from the IEEE Communications Society, as well as the 2001 Texas Instruments DSP and Analog Challenge $100,000 prize.

Isaac Keslassy (M'02, SM'11) received his M.S. and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA, in 2000 and 2004, respectively.

He is currently an associate professor in the Electrical Engineering department of the Technion, Israel. His recent research interests include the design and analysis of high-performance routers and multi-core architectures. The recipient of the European Research Council Starting Grant, the Alon Fellowship, the Mani Teaching Award and the Yanai Teaching Award, he is an associate editor for the IEEE/ACM Transactions on Networking.

Carmi Arad received the B.S. degree in computer engineering (summa cum laude) from the electrical engineering department of the Technion, Haifa, Israel in 1998. He is currently a switching products architect in Marvell Semiconductors, Israel, where he developed packet processors and switching fabrics. His main fields of interest are switch architecture and in particular fast packet lookup, packet counting, traffic management and QoS engines. He issued 19 patents.

Tal Mizrahi is a feature definition architect at Marvell, and a PhD student at the Technion, Israel Institute of Technology. With over 10 years of experience in networking, networking security, and ASIC design, Tal has served in various positions in the industry, including a system engineer, a team leader and an Architect. Tal is an active participant in the Internet Engineering Task Force (IETF), and in the IEEE 1588 working group. Tal received his B.S. and M.S. in Electrical Engineering from the Technion.

Yoram Revah received the B.S. (cum laude) and M.S. degrees in Electrical Engineering from the Technion, Israel in 1997 and 2001, respectively. He worked as a VLSI Engineer and a research assistant in the Electrical Engineering department at the Technion until 2003. He received his Ph.D (summa cum laude) from the Communication Systems Department at Ben Gurion University of the Negev, Israel in 2008, where he held a Kreitman Foundation Fellowship. Currently, he is working in Marvell Israel in a position of Research & Academic Relationship engineer.

Avinatan Hassidim received his doctoral degree from the Hebrew university at 2009. He was later a postdoc at MIT, and is now a senior lecturer at the department of computer science at Bar Ilan university. His primary research interest in algorithms, focusing on distributed solutions and computer networks.